Mesh Denoising with Facet Graph Convolutions – Supplementary Material –

Matthieu Armando, Jean-Sébastien Franco, and Edmond Boyer

In this document, we present additional results to illustrate our method, as well as further experiments to validate our design choices.

1 Raw estimated normals

We visualize here the raw output of our network. For the following figures (1, 2 and 3), we show the noisy input mesh, with faces colored by their normal, the noisy mesh with faces colored by the corrected normal inferred by our network, and the final denoised mesh, with faces colored by their normal. It is interesting to note that our network seems to learn some form of spatial consistency, even though the network returns a single output per facet, and its traning cost is applied per face. This is only made possible because of its convolutional nature. Also, figure 3 reveals the artifacts introduced by the vertex updating step for the *Kinect Fusion* dataset.

2 Experiments on convolutional layers

The key concept behind FeaStNet is that the learned features steer the assignment of filters at each layer. We try a different approach here, regarding the assignment function of equation (7) of the main paper. For our convolution layers, we concatenate position information to the input (*e.g.* received from a previous layer), and our q_m becomes

$$q_m(\mathbf{x}_i, \mathbf{x}_j) \propto \exp(\mathbf{u}_m^\top \mathbf{x}_i + \mathbf{v}_m^\top \mathbf{x}_j + \mathbf{s}_m^\top (\mathbf{p}_j - \mathbf{p}_i) + \mathbf{c}_m) \quad (1)$$

where $\mathbf{p}_k$ denotes the spatial position of node k. That is, filter assignment depends on the input features on i and j, and on the relative spatial position of both nodes. Alternatively, $\mathbf{p}$ could be any signal that gives some measure of the shape of the input graph $\mathcal{G}$. The idea behind this is to decorrelate the signal on $\mathcal{G}$ that we want to process (multiplied throughout the network), and the space that $\mathcal{G}$ lives in (used for assignment only). The same measure $\mathbf{p}$ is used consistently for each layer throughout the network. In order to propagate the position information of nodes to the coarser levels of our achitecture, we perform average pooling on the position of nodes.

We compare the performance of different networks trained for the same number of iterations (60k). Network A is our standard method, trained with the original FeaStNet assignment and both normal and position information as input. Network B has the same architecture, but only normals as input. Network C only has normals as input, but the nodes' positions are used through all layers of the network, to help steer convolutions, on top of the features (as in equation 1).

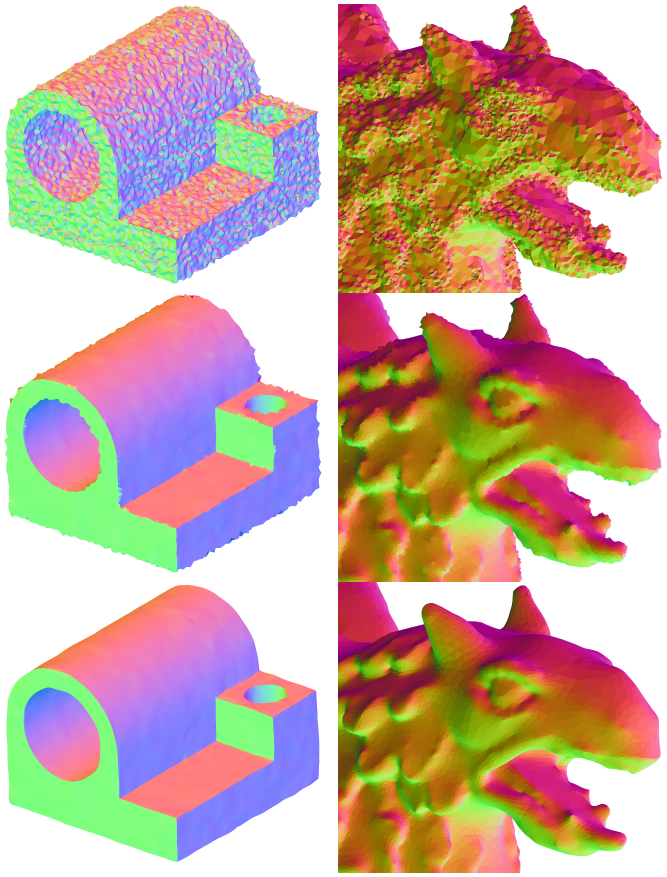


Fig. 1. Examples from the synthetic dataset (with highest noise level). From top to bottom: raw normals, estimated normals, final normals after the vertex updating step.

Figure 4 shows our results on the synthetic dataset. It seems using positions on top of normals increases the performance. More importantly, FeaStNet outperforms our suggested change, which validates their approach of learning assignments that depend on the features rather than on some spatial measure. To push the analysis further, we test network A on meshes with a random rotation applied to all nodes' positions (but not to the nodes' normals). We

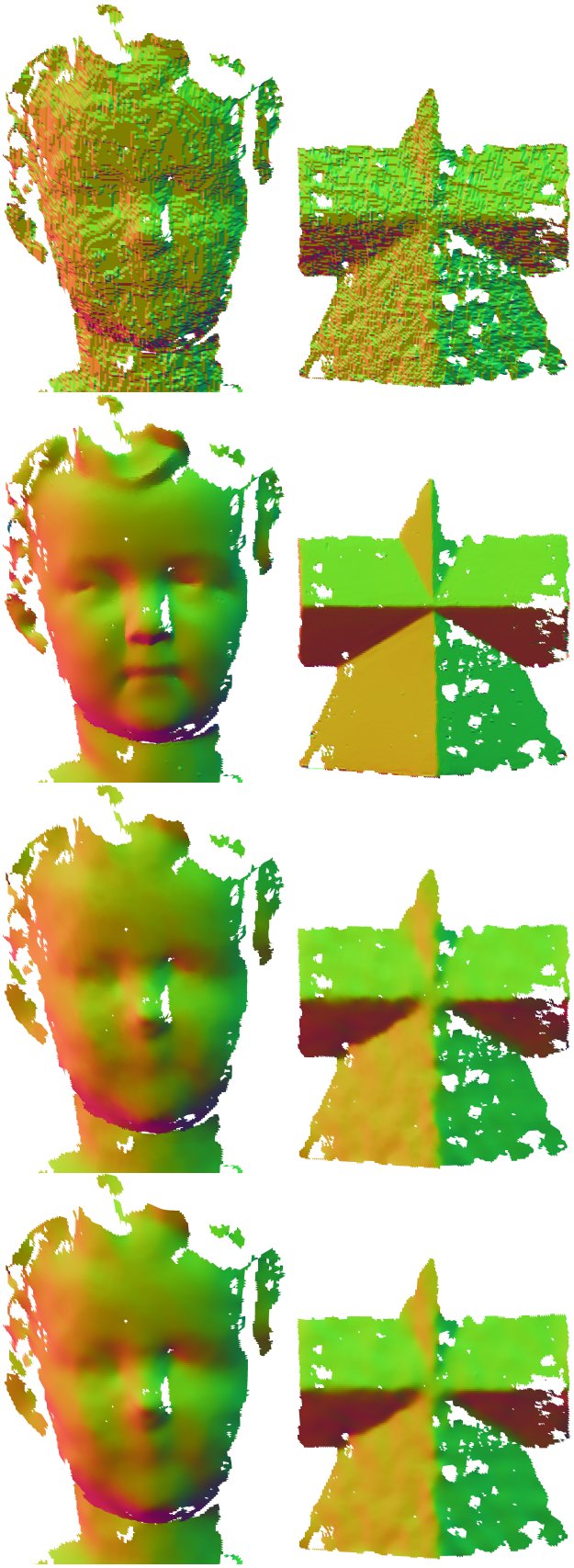


Fig. 2. Examples from the *Kinect v1* dataset. From top to bottom: Noisy mesh colored with: (1) noisy input normals, (2) ground truth normals, (3) estimated normals. (4) denoised mesh after the vertex updating step.

observe no degradation of performance. This suggests that position information is mainly used in an indirect manner by the network, for filter assignment, but not directly for the computation of the output normals. This would mean our insight for network C was sensible, but the network is able to learn it by itself without hard constraint, and more efficiently. It learns how to denoise a signal on a graph with no real notion of geometry.

3 Experiments on FNDs

We wish to test the ability of our network at learning meaningful features that capture the local shape. To this extent, we use the hand-crafted *bilateral filtered facet normal descriptors* (B-FNDs) of [1] as baseline.

We train a network with B-FNDs as input, using the default parameters of [1], which makes up a total of 30 input channels. We compare it to our standard approach using normals only, on the synthetic dataset. Figure 5 shows the performance of both networks on the test set, at arbitrary checkpoints during training. We can observe the FND network converges more quickly to a good solution, as can be expected, since it has access to relevant features at different scales right from the start. As training time increases, both network converge to (and oscillate around) a good solution, and the performance gap is reduced, which shows that our network is able to extract meaningful features from the raw normals. However, the FND network still performs better than the normals one. This might be explained in part by the fact that FNDs indirectly provide information about the *spatial* neighbourhood of faces, which would be hard to simulate for our network based on *connectivity*.

4 Tables

We report in tables 1, 2 3 and 4 the precise numerical results shown in the bar graphs of the main paper. As stated in the paper, results for BMD [2], BNF [3], GNF [4], L0 [5] and Bayesian method [6] are taken from [1].

TABLE 1
Average angular error (in degrees) over test meshes of the synthetic dataset of [1]. Parameters are as follows: BMD: 15 iterations; BNF: $(\sigma_s = \bar{l}_e, \sigma_r = 0.35)$; GNF: $(\sigma_s = \bar{l}_e, \sigma_r = 0.45)$; L0: $(\lambda = 2\lambda_0)$; Bayesian method: $(\sigma = 2\bar{l}_e)$; NLLR: $(\sigma_M = 0.39, v_{iter} = 7, N_k = 7)$, where $\bar{l}_e$ is the average edge length of each mesh.

Method	CAD	Smooth	Features	Total
BMD [2]	7.1082	6.1993	7.9087	7.5148
BNF [3]	4.8016	4.5405	7.2424	6.3210
GNF [4]	5.5006	5.4637	7.7869	6.9470
L0 [5]	5.7363	6.9572	8.4833	7.6152
Bayesian method [6]	5.4276	6.5996	9.1907	7.9458
CNR [1]	4.3645	3.9378	5.6939	5.1604
NormalF-Net [7]	4.7976	3.7995	5.4482	5.1062
NLLR [8]	5.2535	3.8634	5.9396	5.4117
Ours (raw normals)	3.9272	3.8619	5.3201	4.8036
Ours (final result)	**3.5354**	**3.3017**	**4.8507**	**4.3438**

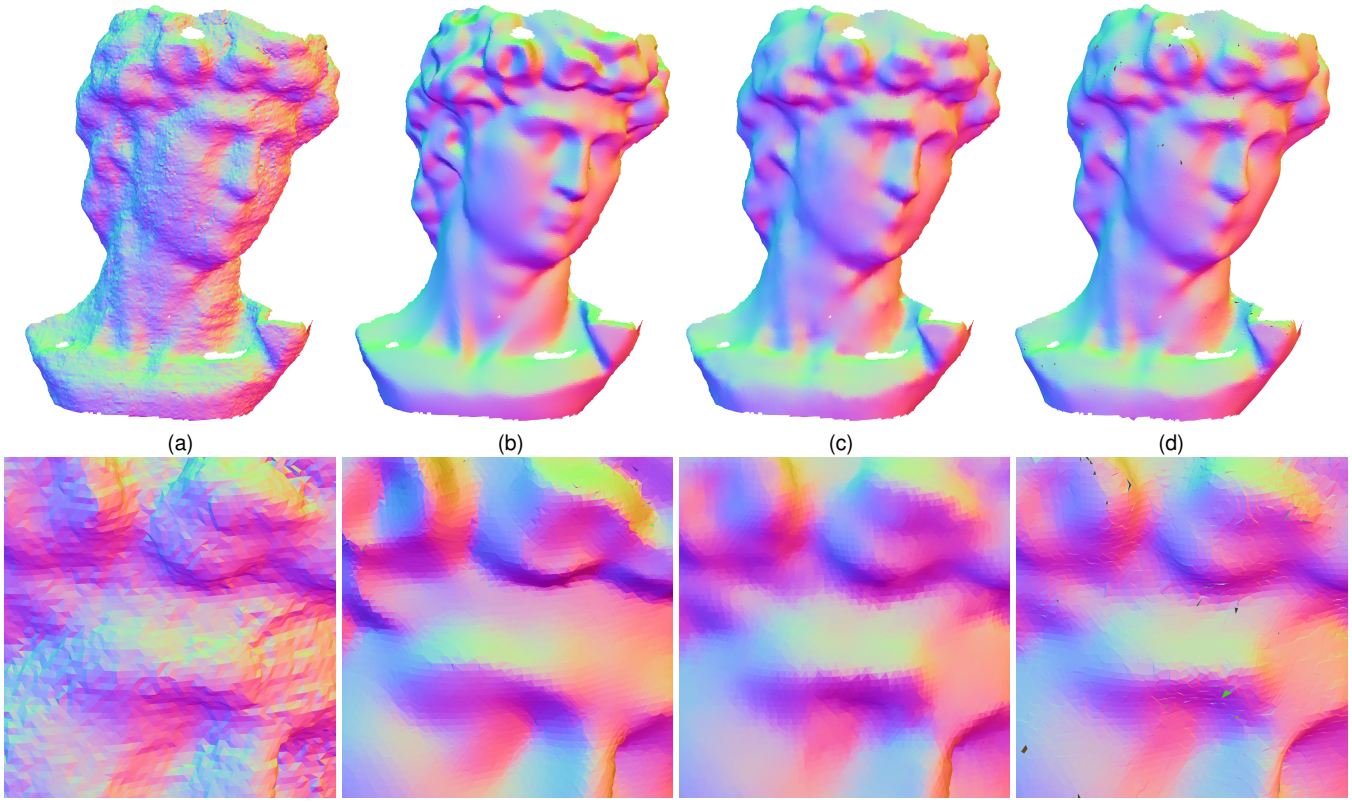


Fig. 3. Examples from the *Kinect Fusion* dataset. **Top**: From left to right: noisy mesh with: (a) noisy input normals, (b) ground truth normals, (c) estimated normals. Far right (d): denoised mesh after the vertex updating step. **Bottom**: close-up view.

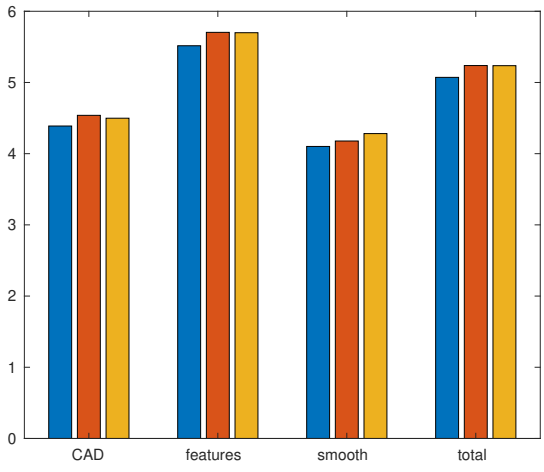


Fig. 4. Comparison of network performance for different choices of input and convolution layers. Average angular error on the synthetic dataset after a fixed number of training steps. **Blue**: Network A (standard). **Red**: Network B (normals only as input). **Yellow**: Network C (position used for assignment only).

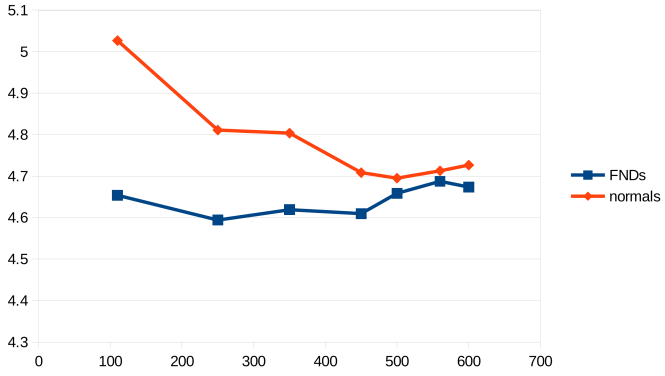


Fig. 5. Comparison of network performance with respect to training time, depending on input features: Average angular error on the test set are plotted against the number of training iterations (in thousands).

5 Additional Figures

In figure 6 we visualize the activation of random convolutional filters in different layers. For the first one or two layers, most channels seem to encode a specific local orientation. For the last layer, a given channel seems to activate for a specific orientation of the denoised normals. Intermediate features are harder to interpret. Figure 7 illustrates the three levels of artificial gaussian noise used in the synthetic dataset of Wang *et al*. [1]. Figures 8 9 and 10 show results on other data from [9] and [10]. Figure 11 shows heatmaps of angular error, and illustrates how the refinement step can smooth out some of the noise. Figure 12 is another comparison showing heatmaps of angular errors.

References

[1] P.-S. Wang, Y. Liu, and X. Tong, "Mesh denoising via cascaded normal regression," *ACM Transactions on Graphics,*

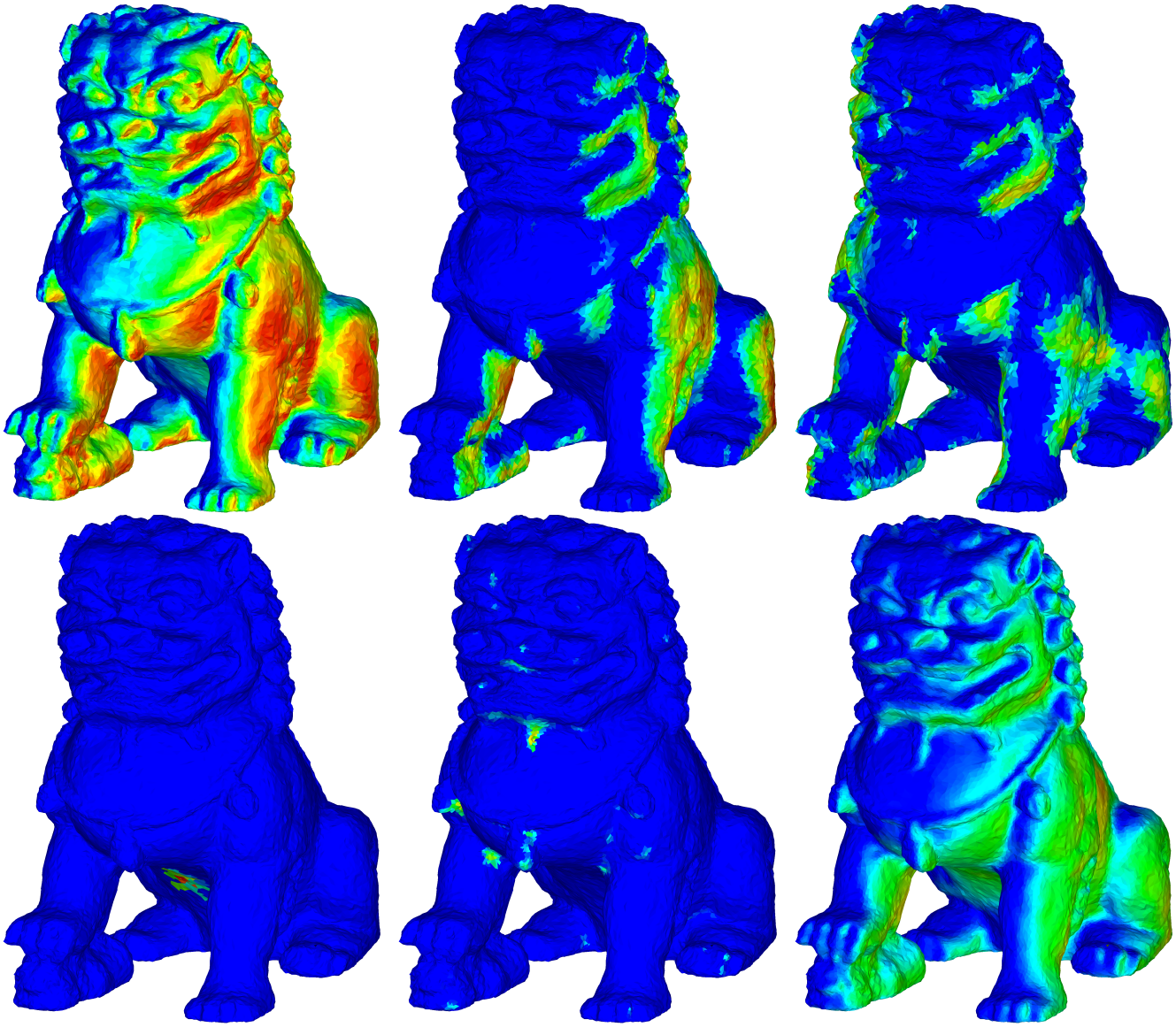


Fig. 6. Examples of activation maps of a random channel, taken from different layers, in order of increasing depth.

TABLE 2
Average angular error (in degrees) over test meshes of the *Kinect v1* dataset of [1].

Method	Boy	Cone	Girl	Pyramid	Total
CNR [1]	8.4380	8.8004	11.4352	8.1194	9.3744
NormalF-Net [7]	9.0732	9.4634	11.9076	9.2291	10.0155
Ours (raw normals)	8.2692	8.4675	10.9106	8.7824	9.1740
Ours (final result)	**7.8747**	**8.2437**	**10.7813**	**8.4489**	**8.8855**

TABLE 3
Average angular error (in degrees) over test meshes of the *Kinect v2* dataset of [1].

Method	Boy	Cone	Girl	Pyramid	Total
CNR [1]	7.8637	6.3333	9.3196	6.8048	8.0381
NormalF-Net [7]	8.2399	6.5979	9.9878	7.1228	8.4873
Ours (raw normals)	8.0465	5.7291	8.9874	6.3396	7.9036
Ours (final result)	**7.7569**	**5.4174**	**8.7833**	**5.8755**	**7.6175**

vol. 35, no. 6, pp. 1–12, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2980179.2980232

[2] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," in *ACM Transactions on Graphics*, vol. 22, no. 3. New York, New York, USA: ACM Press, 2003, p. 950. [Online]. Available: http://portal.acm.org/citation.cfm?doid=882262.882368

[3] Y. Zheng, H. Fu, O. K. C. Au, and C. L. Tai, "Bilateral normal filtering for mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 10, pp. 1521–1530, 2011.

[4] W. Zhang, B. Deng, J. Zhang, S. Bouaziz, and L. Liu, "Guided Mesh Normal Filtering," *Computer Graphics Forum*, vol. 34, no. 7, pp. 23–34, 2015.

[5] L. He and S. Schaefer, "Mesh Denoising via L0 Minimization," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 64:1—-64:8, 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2461965

[6] J. R. Diebel, S. Thrun, and M. Brünig, "A Bayesian method for probable surface reconstruction and decimation," *ACM Transactions on Graphics*, vol. 25, no. 1, pp. 39–59, 2006.

[7] Z. Li, Y. Zhang, Y. Feng, X. Xie, Q. Wang, M. Wei, and P.-A. Heng, "NormalF-Net: Normal

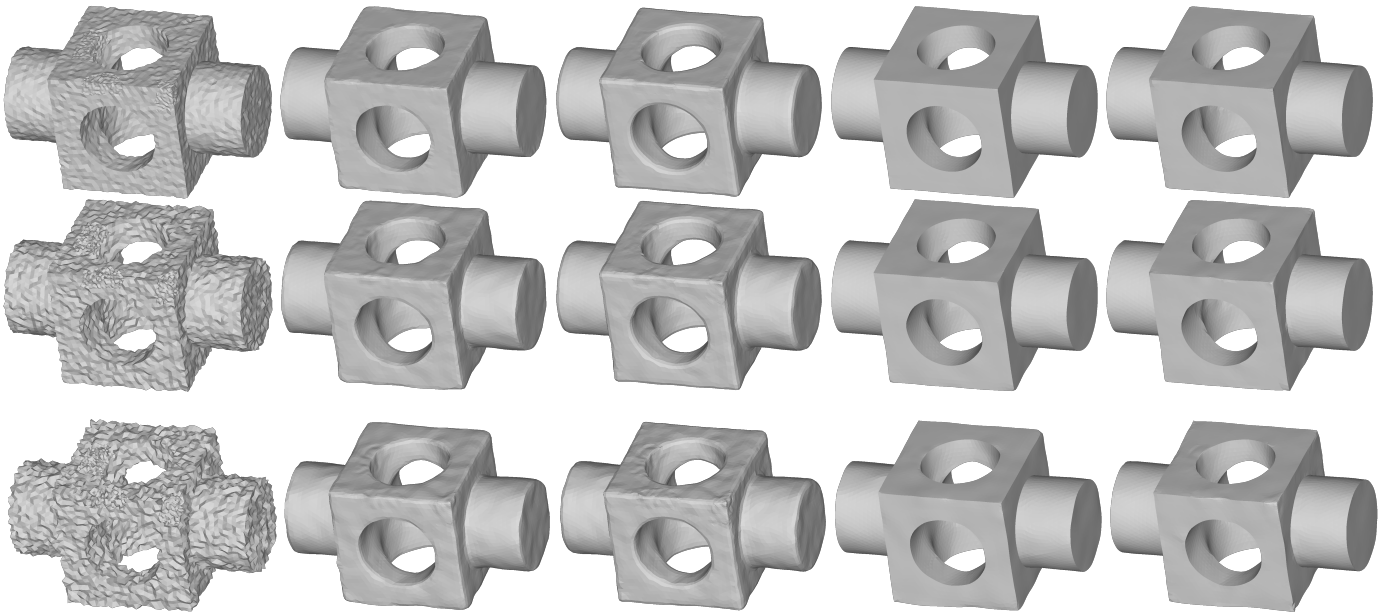


Fig. 7. Examples showing the three different noise levels used in the synthetic dataset of [1]. From top to bottom: increasing noise level.

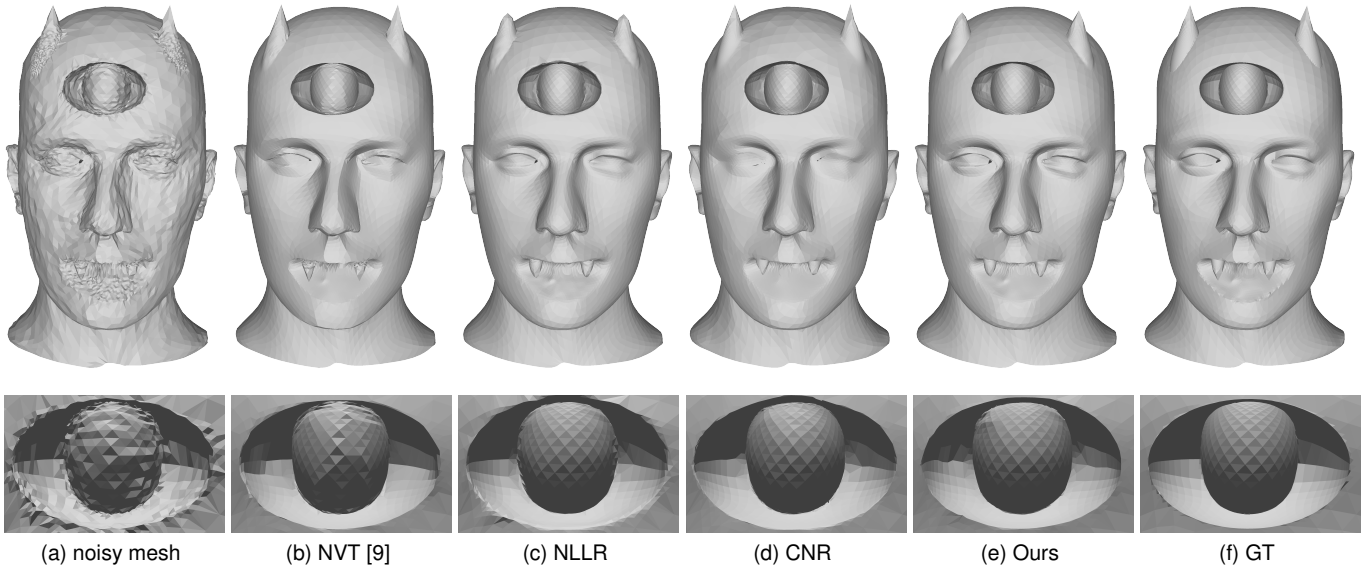


Fig. 8. Results on synthetic data from Yadav *et al*. [9]. Top: whole mesh. Bottom: close-up view of the forehead with grazing light.

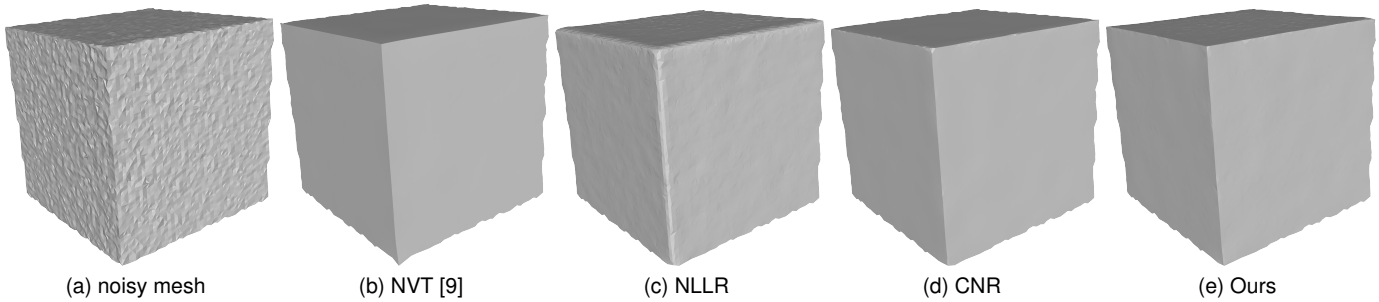
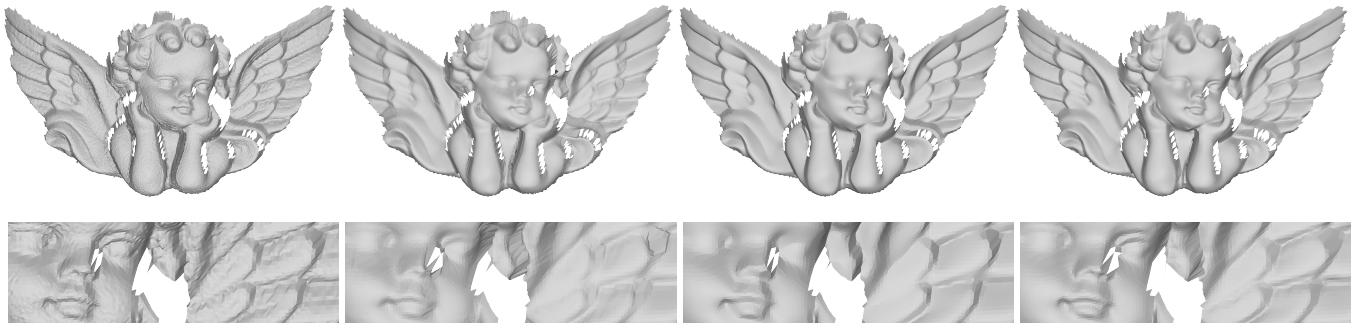


Fig. 9. Results on real scanned data from Yadav *et al*. [9].

filtering neural network for feature-preserving mesh denoising," *Computer-Aided Design*, p. 102861, may 2020. [Online]. Available: https://doi.org/10.1016/j.cad.2020.102861

https://linkinghub.elsevier.com/retrieve/pii/S0010448520300543

[8] X. Li, L. Zhu, C. W. Fu, and P. A. Heng, "Non-Local Low-Rank Normal Filtering for Mesh Denoising," *Computer Graphics Forum*,



(a) noisy mesh (b) NLLR (c) CNR (d) Ours

Fig. 10. Results on real scanned data from Zhang *et al*. [10]. Second row: close-up view

TABLE 4
Average angular error (in degrees) over test meshes of the *Kinect Fusion* dataset of [1]. Parameters are as follows: NLLR: $(\sigma_M = 0.39, v_{iter} = 10, N_k = 10)$

Method	Boy01	Boy02	David	Pyramid	Total
CNR [1]	9.0990	7.6295	**11.9396**	7.4876	9.0427
NormalF-Net [7]	9.2543	7.7595	11.9256	7.7539	9.1698
NLLR [8]	11.5001	9.4517	13.4784	10.0239	11.1005
Ours (raw normals)	**8.8144**	**7.3315**	12.1337	**6.9132**	**8.8160**
Ours (final result)	9.6855	8.0402	12.6370	8.2161	9.6224

vol. 37, no. 7, pp. 155–166, 2018.

[9] S. K. Yadav, U. Reitebuch, and K. Polthier, "Mesh Denoising Based on Normal Voting Tensor and Binary Optimization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 8, pp. 2366–2379, 2018.

[10] W. Zhang, B. Deng, J. Zhang, S. Bouaziz, and L. Liu, "Guided Mesh Normal Filtering," *Computer Graphics Forum*, vol. 34, no. 7, pp. 23–34, 2015.

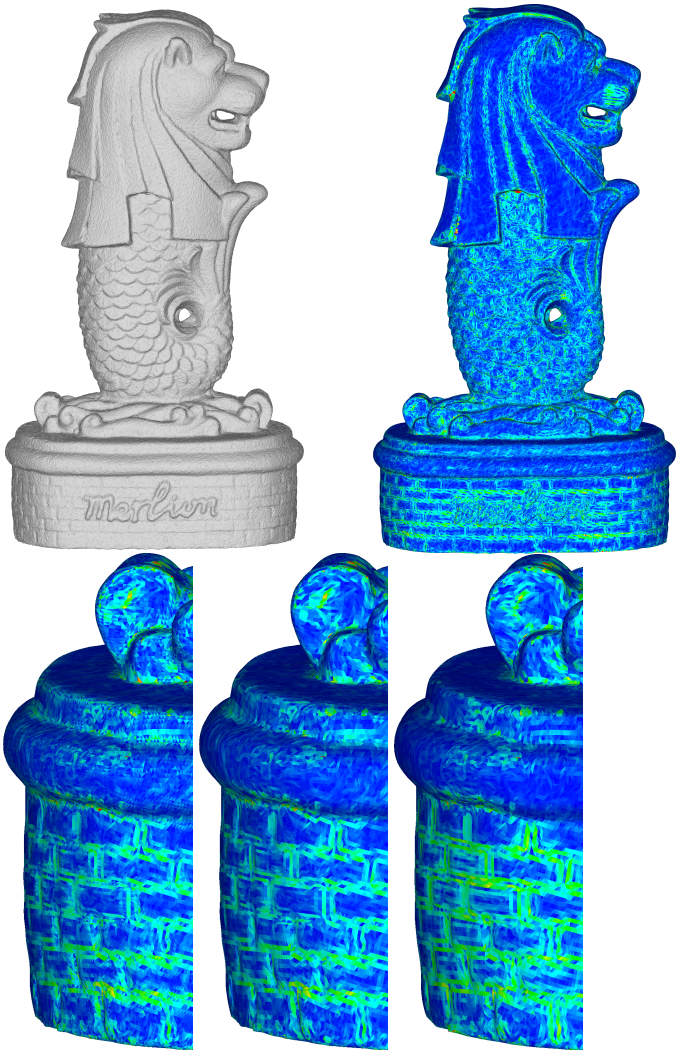
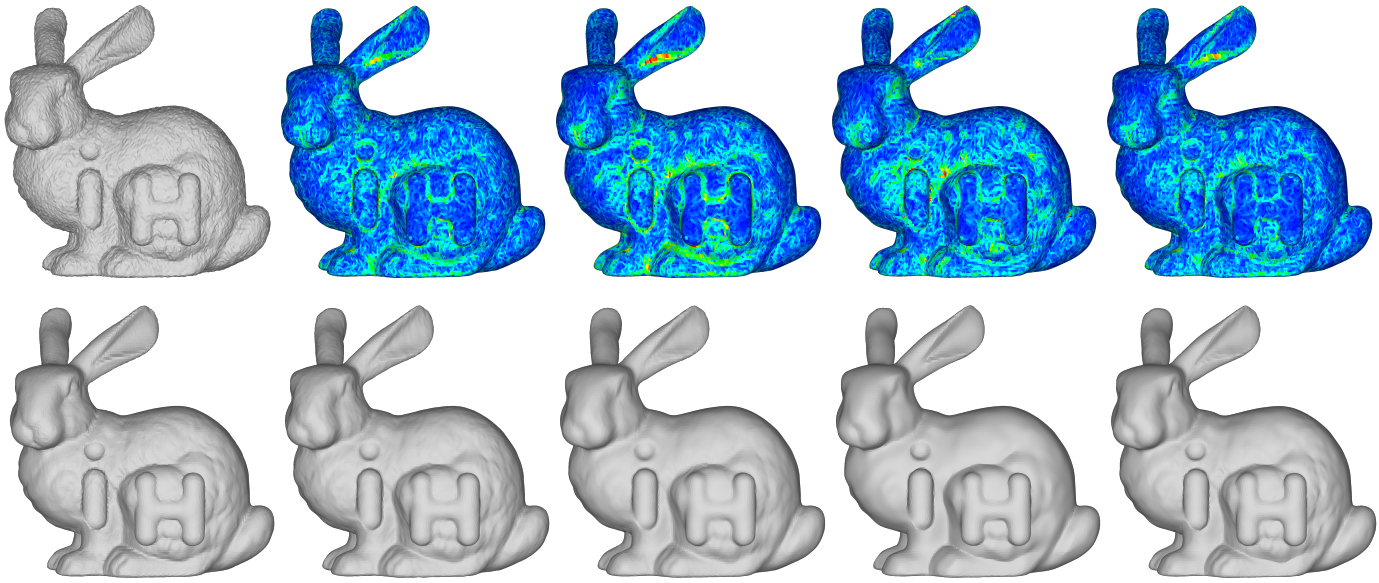


Fig. 11. **Top row**: Left: a noisy mesh. Right: heatmap of angular errors of estimated normals for each face.
Bottom row: Close-up comparison. From left to right: raw output normals of our GCN, normals of the final denoised mesh (with refinement), and normals of CNR (with refinement). The vertex refinement step contributes to the denoising process by smoothing out some of the remaining noise. CNR seems to perform better on smooth surfaces (*e.g*. lion head), whereas our approach seems more precise on complex features (*e.g*. scales and brick pattern).



(a) noisy-GT mesh (b) NormalF-Net (c) NLLR (d) CNR (e) Ours

Fig. 12. Results on the *bunny hi* mesh of the synthetic test set, showing heatmaps of angular errors .